Politecnico di Torino

Matteo Busnelli S273820

Corso di laurea in Ingegneria Gestionale

Classe L8 – Ingegneria dell'informazione

A.A. 2021/2022

Sessione di laurea ottobre 2022

Applicazione per la simulazione della Superlega

Relatore: Prof. Fulvio Corno

INDICE

PROPOSTA DI PROGETTO

Studente proponente

S273820 Matteo Busnelli

Titolo della proposta

Simulazione Superlega

Descrizione del problema proposto

Vorrei elaborare un'applicazione che permetta all'utente di scegliere una delle 12 squadre facenti parte alla superlega, e di "crearsi" la rosa di giocatori da lui preferita, nel rispetto dei vincoli di :

- Numero minimo di giocatori per ruolo
- Stipendi totali non superiori al salary cap

L'utente, per poter effettuare al meglio la sua scelta, potrà accedere a tutte le statistiche/informazioni di ogni giocatore.
Una volta composta la squadra, potrà simulare l'intero campionato e vedere in che posizione si è classificata la sua squadra, in base ai giocatori scelti, i loro valori di attacco/difesa, tenendo anche conto della probabilità di infortuni e altre casistiche.
Le altre squadre, non scelte dall'utente, verranno composte di giocatori, in maniera casuale, nel rispetto dei vincoli imposti. Alla fine, si otterrà la squadra vincitrice.

Descrizione della rilevanza gestionale del problema

Si tratta di un problema di rilevanza gestionale in quanto l'utente deve immedesimarsi nel ruolo di manager di una squadra di calcio, gestendo al meglio gli stipendi dei giocatori da inserire in rosa, per vincere il campionato.

Il progetto della superlega è un tema estremamente discusso negli ultimi mesi. Si tratta di un nuovo modello di calcio, non entrato ancora in vigore, che si rifà al modello americano dell'NBA. Infatti, cambia radicalmente la visione della squadra, in quanto viene vista come una vera e propria franchigia, all'interno di un mercato internazionale. Si tratta di un progetto molto promettente, in quanto il moderno calcio europeo, soprattutto dopo la crisi economica subita durante la pandemia e tutt'ora in atto anche a causa della guerra, ha bisogno di una nuova scossa e un nuovo ridimensionamento; infatti, sono sempre di più le società indebitate (vedi ad esempio il Barcellona) che vedono anche nell'aspetto sportivo, grosse ripercussioni. L'attuale calcio europeo non porta più grossi introiti, si tratta oramai di un dato di fatto, la UEFA Champions League sta perdendo sempre di più l'interesse e il fascino che l'ha sempre contraddistinta, sicuramente anche per l'accesso a tale competizione di un numero sempre maggiore di squadre sconosciute, che dal punto di vista economico, non portano alcun interesse.
Quindi, l'idea della superlega è quello di permettere l'accesso ad un numero limitato di squadre, si parla dei top club europei, al fine di garantire un grande interesse collettivo ad ogni partita, il quale porterebbe a grandi vantaggi economici.

Una delle idee principali della superlega è l'introduzione di un salary cap, in cui il costo di ingaggi, impostato di anno in anno, non può essere superato. L'idea è quindi di creare un campionato (una nuova Champions league) con le migliori squadre europee che non possono spendere più di una certa cifra in ingaggi, al fine di garantire, almeno sulla carta, un certo equilibrio di partenza tra tutti i partecipanti.

Nulla è ancora certo riguardo all'entrata in vigore della superlega, ma ciò che è sicuro è che il mondo del calcio ho bisogno di un ridimensionamento, e qualche provvedimento in questa direzione, sicuramente verrà preso.

Descrizione dei data-set per la valutazione

Statistiche e stipendio giocatori:
https://www.kaggle.com/stefanoleone992/fifa-22-complete-player-dataset
In questo dataset, gli stipendi dei giocatori sono netti e calcolati su base settimanale. Io li convertirò su una base annuale di 10 mesi e lordi.
Squadre:
Creo io un dataset semplicissimo contenente le 12 squadre e qualche informazione su di esse.

Descrizione preliminare degli algoritmi coinvolti

Salvare le scelte dell'utente riguardo ai giocatori in apposite strutture dati.
Gestire e manipolare tali strutture dati per il controllo dei vincoli imposti.
Assegnazione casuale per riempire le altre undici squadre non scelte dall'utente.
Una volta riempite tutte le strutture dati, simulazione ad eventi in cui si simula l'intero campionato in modalità play-off.

Descrizione preliminare delle funzionalità previste per l'applicazione software

L'utente potrà impostare, scegliendo tra una serie di valori, il salary cap e, basandosi su tale valore, scegliere la squadra che vuole comporre nel rispetto di tutti i vincoli. Per scegliere i giocatori, potrà valersi di tutte le statistiche necessarie, ossia in base a valori ipotetici assegnati al singolo giocatore da parte degli sviluppatori di fifa, i quali dovrebbero rispettare le reali capacità del giocatore in questione. Una volta composta la squadra, l'utente potrà simulare l'intero campionato per vedere il posizionamento in classifica della squadra da lui composta.

Ulteriori spiegazioni a seguito della risposta del Prof. Corno:

Stavo pensando di rendere più interessante la parte iniziale, ossia quella di scelta dei giocatori nella squadra. L'obiettivo è quello di crearsi una squadra il più ottimizzata possibile che permetta quindi di scegliere i giocatori migliori e più adatti alla squadra che si sta creando.

La squadra è un grafo di 11 vertici che corrispondono ai giocatori (il modulo può variare in base alla scelta dell'utente), ogni vertice è collegato con i suoi vicini e il peso dell'arco è dato dall'intesa reciproca tra i due giocatori. L'intesa è calcolata in base a se i due giocatori in questione sono della stessa nazionalità, se giocano nello stesso campionato o nella stessa squadra. Il valore di intesa è compreso tra 0 e 3 (0 se non è rispettata nessuna delle tre condizioni, 1 se è rispettata solo una, e così via). L'utente non è vincolato a un minimo valore, può comporsi la squadra liberamente, sarà il programma stesso ad indicare l'attuale stato, il quale, a valori crescenti, garantirà una probabilità crescente di vittoria; infatti, l'intesa totale sarà un parametro influente all'interno della simulazione.

Inoltre, un ulteriore possibilità per crearsi la squadra migliore sta nello scegliere i giocatori migliori (con overall più alto e stipendio più basso per garantirsi un buon giocatore che non vada ad impattare eccessivamente sul monte ingaggi) quindi, in base a valori crescenti di questo rapporto, il giocatore sarà "migliore" di un altro e di conseguenza permetterà di crearsi una squadra più vincente. L'utente in fase di scelta dei giocatori, potrà effettuare delle ricerche specifiche in base ai parametri a cui è interessato, potrà farsi suggerire dal programma stesso il giocatore consigliato da inserire in rosa e, se si tentasse di inserire un nuovo giocatore "migliore" rispetto ad un altro già scelto ed è stato raggiunto il limite salariale, quest'ultimo giocatore potrà essere rimosso. Il programma suggerirà quindi allo stesso tempo sia il giocatore migliore da acquistare, sia il giocatore peggiore attualmente presente in rosa che può essere scartato. Il giocatore migliore sarà quindi il giocatore con rapporto overall/stipendio più alto scelto all'interno dei criteri che garantiscano un livello di intesa accettabile (almeno 1).

La simulazione si basa sull'analisi e la combinazione di parametri noti, innanzi tutto l'intesa rappresenta un fattore moltiplicativo della probabilità di vittoria. In generale, tutte le statistiche presenti nel dataset di ogni singolo giocatore, contribuiscono al risultato finale, inoltre prevedono degli algoritmi di randomizzazione che permettono quindi l'inserimento nella coda degli eventi di possibilità di infortunio, espulsioni, ecc. tali probabilità dipendono sempre dalle statistiche del singolo giocatore (un giocatore che es. 60 di fisico, avrà più probabilità di infortunarsi rispetto ad uno con 90).

Fondamentale è l'equilibrio della squadra nei quattro ruoli principali (portiere, difesa, centrocampo e attacco). Per ogni ruolo di ogni squadra verrà calcolato un valore complessivo dei giocatori in quel ruolo, e confrontato con un valore complessivo del ruolo che si oppone della squadra avversaria (es. SQUADRA A, attacco 80, SQUADRA B difesa 70, nella durata complessiva della simulazione, l'attacco della SQUADRA A prevale sulla difesa della SQUADRA B, si andrà a vedere negli altri ruoli quale delle due squadre prevale e verrà decretato il vincitore). Verranno inoltre mostrate in output all'utente tutte le statistiche della partita.

DESCRIZIONE DETTAGLIATA DEL PROBLEMA AFFRONTATO

Negli ultimi anni, il mondo dello sport ha assunto un ruolo sempre più centrale all'interno dell'economia mondiale. Infatti, le più grandi società sportive odierne sono delle vere e proprie aziende, quotate in borsa, che fatturano centinaia di miliardi di euro ogni anno.

A causa della crisi dovuta alla pandemia, sempre più realtà sportive hanno riscontrato ingenti perdite economiche, a tutto ciò vanno correlati i numerosi problemi che oramai riguardano da anni la UEFA Champions League, dovuti principalmente agli elevatissimi diritti tv.

In generale sono state evidenziate evidenti perdite economiche che hanno portato il mondo del calcio alla necessità di un rinnovamento per dare una nuova 'scossa' all'intero movimento.

Per questi, e altri numerosi motivi, è stato ideato il progetto della Superlega.

La Superlega sarà - nel caso vada in porto il progetto - una competizione calcistica privata (ovvero fuori dal dominio di Fifa, Uefa e delle leghe nazionali) a cadenza annuale tra i più blasonati club d'Europa.

Si tratta di una competizione per club alternativa alla Champions League che riunisca le migliori squadre europee in una sorta di campionato di super élite: è un progetto al quale da diverso tempo stavano lavorando alcuni dei più influenti presidenti del calcio continentale, guidati da Florentino Perez. Al punto da portare la Uefa a minacciare una causa milionaria contro tutti i club che vorranno traslocare dalle competizioni ufficiali (Champions ed Europa League) ed entrare a far parte della superlega e i loro giocatori l'esclusione dalle Nazionali e da tutte le competizioni Uefa e Fifa.

Al centro di tutto ci sono, semplicemente, i soldi. Secondo le stime, le 12 società aderenti al progetto Superlega riceverebbero subito 350 miliardi di euro da spartire equamente tra loro, una vera boccata di ossigeno per realtà che, causa crisi Covid-19, hanno perso decine di miliardi di fatturato.

La Superlega si rifà al modello americano dell'NBA, ovvero sistema in cui ogni anno viene fissato un limite salariale, o salary cap, che non può essere superato con la somma degli stipendi di tutti gli addetti ai lavori (giocatori e non). Oltre ai motivi economici, il progetto Superlega va a garantire, almeno sulla carta, un maggiore equilibrio iniziale nel campionato. Sicuramente porta un numero di spettatori maggiore ogni partita, in quanto si avrebbero confronti tra le top squadre europee settimanalmente.

Il progetto Superlega è attualmente stato bloccato dalle minacce provenienti da UEFA e FIFA. Al momento non si hanno notizie riguardanti l'effettiva entrata in vigore o meno nel futuro prossimo.

Per quanto riguarda l'applicativo realizzato, l'utente potrà scegliere oltre alla squadra e il modulo preferito, il salary cap che dovrà essere rispettato durante tutta la fase di composizione della squadra.

Il programma suggerirà dei giocatori che garantiscono buone prestazioni (overall alti) e che non hanno uno stipendio che influisca eccessivamente all'interno del totale degli stipendi della squadra (stipendi bassi).

Per ogni giocatore scelto dall'utente per la sua squadra, contemporaneamente le altre undici squadre sceglieranno un giocatore del medesimo ruolo. In qualsiasi momento l'utente potrà visualizzare le statistiche del giocatore desiderato (statistiche personali, valori in attacco e valori in difesa), e potrà anche rimuovere un giocatore che aveva scelto in precedenza.

Terminata la fase di composizione della squadra, sarà visualizzata una finestra di riepilogo prima di accedere alla parte finale di simulazione del campionato. Quest'ultima è organizzata su quattro turni. Nel turno 1, le dodici squadre si sfidano in sei partite, le vincenti accedono al turno 2. Allo stesso modo, le tre vincenti del turno 2 accederanno alla semifinale in cui verrà stilata una classifica. Le prime due classificate accederanno alla finale. In caso di parità tra i punti, si osserverà la differenza gol. Infine, verrà simulata la finale per decretare il vincitore. In ogni turno della simulazione sarà possibile visualizzare l'elenco di tutti gli eventi che sono avvenuti durante la singola partita presa in considerazione (gol, espulsione o infortunio).

DESCRIZIONE DEL DATA – SET UTLIZZATO PER L'ANALISI

Ho utilizzato un data – set denominato 'fifa22' contenente due tabelle: giocatore e squadra.

- Tabella giocatore:

questa tabella composta da una grande quantità di dati, l'ho trovata al seguente link: https://www.kaggle.com/stefanoleone992/fifa-22-complete-player-dataset

contiene tutte le informazioni sulle quali si basa l'applicativo. A partire dal campo fondamentale dello stipendio di ogni giocatore, per arrivare a tutte le statistiche puramente tecniche che contribuiranno allo svolgimento della simulazione.

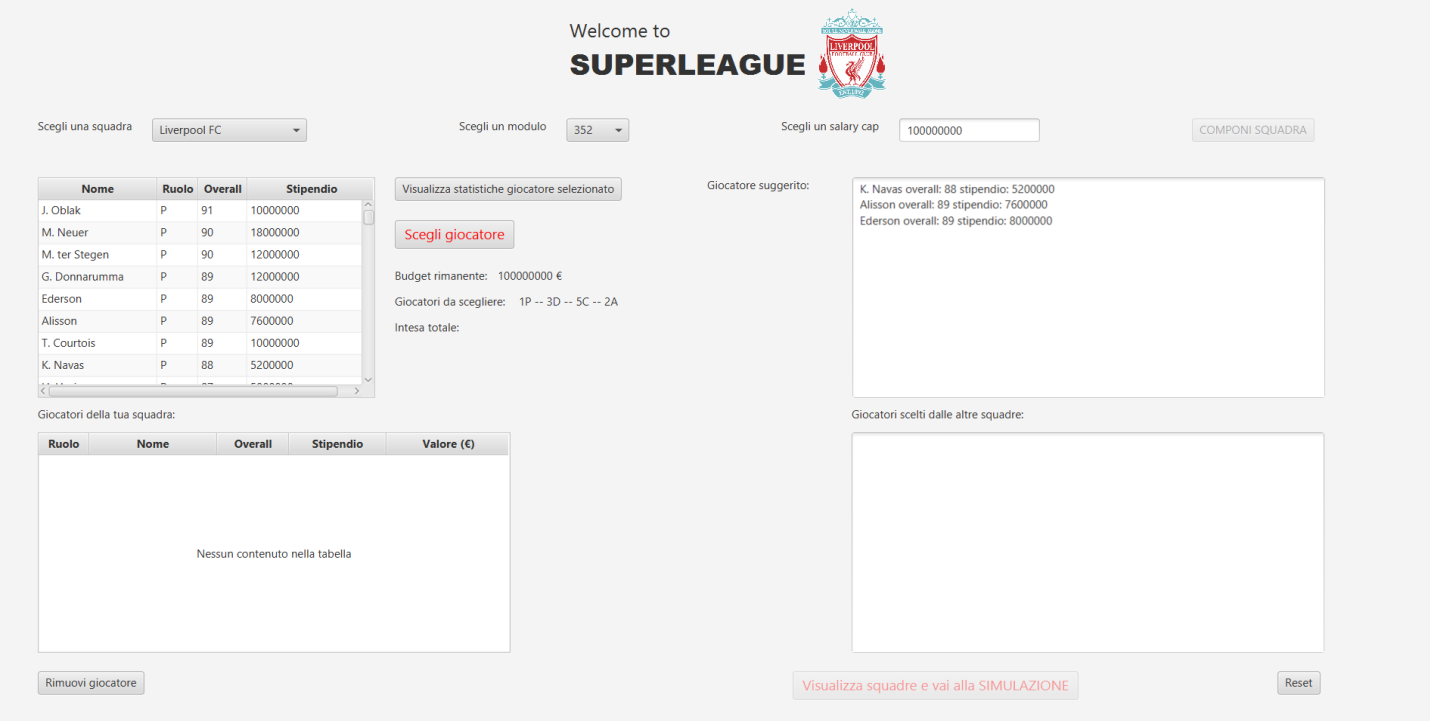
- Tabella squadre:

questa tabella l'ho creata io manualmente e contiene le 12 squadre facenti parte della superlega.

DESCRIZIONE STRUTTURE DATI / ALGORITMI UTILIZZATI

L'applicativo è stato realizzato mediante l'utilizzo del linguaggio Javafx, sfruttando anche SceneBuilder per l'elaborazione delle interfacce grafiche.
Il programma è stato realizzato implementando sia il pattern MVC per garantire una corretta divisione delle operazioni e quindi delegare al *Model* tutta la logica applicativa.
L'applicativo è quindi stato diviso in tre packages: controller, db, model.

All'avvio dell'applicazione, l'utente potrà scegliere la squadra desiderata e il modulo preferito mediante due comboBox, inoltre potrà inserire manualmente nell'apposita textBox il salary cap. Successivamente, dopo aver premuto il bottone 'Componi squadra', potrà scegliere tramite l'apposita table view, il giocatore da lui preferito. L'applicativo gli suggerisce, allo stesso tempo, un elenco dei tre giocatori (del ruolo che l'utente deve scegliere), con rapporto overall/stipendio maggiore; quindi, giocatori 'forti' che hanno uno stipendio basso. Inoltre, sono presenti i contatori del budget rimanente e nel numero di giocatori ancora da scegliere per ogni ruolo. Nella parte bassa è presente una table view che funge da tabella di riepilogo dei giocatori scelti dall'utente. Infine, è presente una text area dove, progressivamente, verranno visualizzati tutti i giocatori scelti dalle altre undici squadre della superlega.



Tutti i giocatori sono contenuti all'interno di una HashMap, sono inoltre suddivisi in ruoli all'interno di più ArrayList. Anche le squadre sono contenute all'interno di un ArrayList.

Al click del bottone 'Componi squadra', la text area dei giocatori suggeriti viene popolata, chiamando un metodo del Model che ritorna una lista dei giocatori suggeriti.

```
public List<Giocatore> giocatoreSuggerito(String ruolo, double budget) {
    List<Giocatore> oro = new ArrayList<>();
    List<Giocatore> argento = new ArrayList<>();
    List<Giocatore> bronzo = new ArrayList<>();
    List<Giocatore> platino = new ArrayList<>();
    List<Giocatore> result = new ArrayList<>();


        for(Giocatore g : this.giocatori.values()) {
            if(g.getRuolo().equals(ruolo) && g.getStipendio()< budget) {

                if(g.getOverall()>=88) {
                    oro.add(g);

                }
                else if(g.getOverall()>=82) {
                    argento.add(g);

                }
                else if(g.getOverall()>=74){
                    bronzo.add(g);

                }else {
                    platino.add(g);
                }

            }


    }

        ordinaPS(oro);
        ordinaPS(argento);
        ordinaPS(bronzo);
        ordinaPS(platino);

        result.addAll(oro);
        result.addAll(argento);
        result.addAll(bronzo);
        result.addAll(platino);

    return result;
```

Dopo aver suddiviso i giocatori in liste diverse in base all'overall (ovviamente verranno suggeriti prima i giocatori con overall maggiore), le rispettive liste verranno ordinate secondo un rapporto overall/stipendio maggiore:

```
private void ordinaPS(List<Giocatore> l) {

    Collections.sort(l, new Comparator<Giocatore>() {

        @Override
        public int compare(Giocatore o1, Giocatore o2) {

            double v1 = (double)o1.getOverall()/o1.getStipendio();
            double v2 = (double)o2.getOverall()/o2.getStipendio();

            if(v1>v2)
                return -1;
            else
                return 1;

        }

    });

}
```

L'utente potrà quindi decidere se ascoltare o meno il suggerimento dell'applicativo andando a scegliere il giocatore che preferisce.

Al click del bottone 'Scegli giocatore', verrà chiamato un metodo del Model che controlla se il giocatore scelto dall'utente, rispetta i requisiti del programma. Se sì verranno settati i contatori e i parametri.

```
public boolean giocatoreScelto(Giocatore g) {
    boolean scelta = false;
    if(this.giocatoriScelti.isEmpty()) {
        //creo il grafo
        this.squadraScelta.creaGrafo();
        for(Squadra s : this.squadre) {
            s.creaGrafo();;
        }
    }
    if(this.budgetRimasto - g.getStipendio() >= 0) {
        String ruolo = g.getRuolo();
        this.giocatoriScelti.add(g);
        this.squadraScelta.setTotAttacco(this.squadraScelta.getTotAttacco()+g.getAttacco().getTotAttacco());
        this.squadraScelta.setTotDifesa(this.squadraScelta.getTotDifesa()+g.getDifesa().getTotDifesa());
        g.setSquadraSuperlega(squadraScelta);
        this.budgetRimasto -= g.getStipendio();
        this.squadraScelta.aggiungiVertici(g);
        //vertici aggiunti solo per mia squadra
        scelta = true;
        squadraScelta.getGiocatoriDellaSquadra().add(g);
        if(ruolo.equals("P")) {
            this.numPortieri--;
            this.portieri.remove(g);
            this.giocatori.remove(g.getFifaId());
        }
        else if(ruolo.equals("D")) {
            this.numDifensori--;
            this.difensori.remove(g);
            this.giocatori.remove(g.getFifaId());
        }
        else if(ruolo.equals("C")) {
            this.numCentrocampisti--;
            this.centrocampisti.remove(g);
            this.giocatori.remove(g.getFifaId());
        }
        else {
            this.numAttaccanti--;
            this.attaccanti.remove(g);
            this.giocatori.remove(g.getFifaId());
        }
    }
    return scelta;
```

Contemporaneamente alla conferma della scelta del giocatore selezionato dall'utente, le altre unidici squadre della superlega effettueranno a loro volta la scelta di un giocatore; questo caso l'ho gestito con un metodo generale, un metodo per trovare il giocatore da assegnare alla squadra i-esima e un metodo per effettuare dei controlli se effettivamente il giocatore trovato nel metodo precedente è valido per la squadra in questione (con annesso un ulteriore metodo per il confronto tra lo stipendio e delle percentuali del budget rimasto di squadra. Da segnalare che il budget totale di squadra, l'ho diviso, secondo certe percentuali, in budget destinato ai portieri, difensori, centrocampisti e attaccanti):

I due metodi più interessanti sono probabilmente la ricerca del giocatore da assegnare e i controlli sul giocatore trovato:

```
private Giocatore trovaGiocatoreDaAssegnare(String ruolo, Squadra squadra) {
    Giocatore g = null;
    boolean giocOk = false;
    int cont = 0;
    List<Giocatore> listaGiocatoriPerSquadre = new ArrayList<>(this.getAllGiocatoriByRuolo(ruolo));

    while(!giocOk) {

        if(cont<listaGiocatoriPerSquadre.size()) {
            g = listaGiocatoriPerSquadre.get(cont);
            cont++;
        }else {
                g = giocatoreCasuale(listaGiocatoriPerSquadre);

        }

        giocOk = controlliGiocatore(g, squadra);
    }


    return g;
}


private boolean controlliGiocatore(Giocatore g, Squadra squadra) {
    String ruolo = g.getRuolo();

    if(ruolo.equals("P")) {
        //un portiere non può costare più del 20% del budget totale
        if(g.getStipendio()> perRuoloModulo(ruolo)*squadra.getBudgetPortiere()) {

        return false;
        }
        else
            return true;
    }
    else if(ruolo.equals("D")){
        //un difensore non può costare più del 30% del budget totale
        if(g.getStipendio()>perRuoloModulo(ruolo)*squadra.getBudgetDifesa()) {

            return false;
        }
        else
            return true;

    }
    else if(ruolo.equals("C")){
        //un centrocampista non può costare più del 70% del budget totale
        if(g.getStipendio()>perRuoloModulo(ruolo)*squadra.getBudgetCentrocampo()) {
            return false;
        }
        else
            return true;
    }
    else {
        //un attaccante può avere lo stipendio che vuole
        if(g.getStipendio()>perRuoloModulo(ruolo)*squadra.getBudgetAttacco()) {
            return false;
        }
        else
            return true;

    }
}
```

Sono presenti, inoltre, più metodi che vanno ad ordinare le liste secondo le necessità, fanno tutti uso di un comparatore.

È presente, un metodo per la ricerca di un giocatore casuale nel caso in cui non venga trovato nessun giocatore adatto, un metodo che permette di rimuovere un giocatore già scelto in precedenza andando quindi ad eliminarlo dalle liste e aggiornando nuovamente i contatori, e un metodo per resettare tutto ciò che è stato svolto fino a quel momento.

Ogni volta che viene scelto un nuovo giocatore, sia dalla squadra scelta dall'utente che da una delle altre undici squadre, esso viene aggiunto al grafo della squadra; i vertici sono appunti tutti i giocatori scelti, gli archi tra tutte le coppie di giocatori, hanno un loro peso, il quale rappresenta l'intesa tra i due giocatori in questione. Il suo valore sarà massimo (=3) se i due giocatori sono della stessa nazionalità, giocano nello stesso campionato e giocano nella stessa squadra; altrimenti esso diminuirà in base a quante delle condizioni appena descritte non sono rispettate. L'intesa aumenterà di una probabilità pari a intesa% di giocare un azione durante la simulazione di una partita.

Sempre in questa prima pagina di composizione della squadra, si potrà accedere ad una nuova pagina, di riepilogo delle statistiche del giocatore selezionato.

Una volta terminata la composizione della squadra, al click del bottone 'Visualizza squadre e vai alla simulazione', si sarà indirizzati ad un nuovo controller di riepilogo dei giocatori scelti da tutte le squadre. Successivamente si potrà accedere alla parte finale del progetto in cui sarà svolta la simulazione.

La simulazione viene suddivisa in quattro turni. Il primo turno è composto da tutte e dodici le squadre, al click del bottone 'Simula turno' verrà simulato l'intero turno. Soltanto le vincenti accederanno al turno successivo. Sarà possibile visualizzare il risultato della partita (in caso di pareggio, il programma calcola il vincitore dopo i calci di rigore). Inoltre, al termine della simulazione del turno, verrà reso visibile un bottone per la visualizzazione degli eventi avvenuti in tutte le partite di quel turno; gli eventi sono contenuti in una lista della classe Statistiche e sono un oggetti di tipo MinEvento.

Le vincenti del secondo turno accedono alla semifinale, in essa verrà calcolata la classifica, soltanto le prime due classificate accedono alla finale. La classifica è ordinata con un algoritmo di ordinamento che si basa prima sui punti totalizzati, in caso di parità osserva la differenza gol:

```
Collections.sort(this.squadreSemifinale, new Comparator<Squadra>() {

    @Override
    public int compare(Squadra o1, Squadra o2) {

        if(o1.getPuntiSemifinale()!=o2.getPuntiSemifinale()) {
            return o2.getPuntiSemifinale()-o1.getPuntiSemifinale();
        }
        else
            return o2.getDiffPunti()-o1.getDiffPunti();
    }
});
```

La finale verrà simulata esattamente come le altre partite.

Fondamentale risulta quindi la classe simulatore, la quale riceve come parametri sempre e solo due squadre (la simulazione avviene per turni, ma la classe simulatore simula una partita alla volta). Essa ritorna un oggetto Statistiche, contente tutte le statistiche della partita tra le due squadre passate come parametro.

Il numero di azioni di una partita è arbitrario: fissando un intervallo di tempo pari a 5 minuti, se consideriamo una partita durare in totale 90 minuti, possiamo ottenere quindi 18 slot temporali. In questi 18 slot, non sempre si verifica un evento. Infatti, supponendo che in media avvengono 10 eventi per partita, la probabilità che in uno slot specifico avvenga un evento è pari a 10/18. In base a questa probabilità, si otterrà quindi il numero di azioni 'importanti' giocate nel corso della partita. In questo modo ogni partita avrà un numero di azioni diverso e non fisso. Tutto ciò è stato implementato nel seguente modo all'interno del metodo init() del simulatore:

```
int cont=0;
while(cont<this.slotTemporali) {
    double prob = Math.random();

    if(prob<=this.probEvento) {
        this.totAzioni++;

    }
    cont++;
}
```

L'intera simulazione si basa su un parametro fondamentale, chiamato 'rapportoForza': è un parametro che divide la somma totale dell'attacco più la difesa della squadra di casa per la somma totale dell'attacco più la difesa della squadra in trasferta. Se questo parametro è circa pari a 1, le due squadre possiamo assumerle più o meno equivalenti in termini di forza, avranno quindi la stessa probabilità di giocare un'azione (e quindi giocando un azione di fare gol). Se questo rapporto è invece >2 o <0.5, significa che rispettivamente la squadra di casa o la squadra in trasferta sono il doppio "più forti" dell'avversaria, avranno quindi il doppio di probabilità di giocare un azione. Tali probabilità sono aumentate (o diminuite) dal parametro 'intesa' descritto precedentemente. Nella classe Simulatore, esso viene calcolato come la differenza tra l'intesa della squadra di casa e l'intesa della squadra in trasferta. Se >0 aumenta la probabilità di far giocare un'azione alla squadra di casa (perché vuol dire che intesaCasa > intesaTrasferta), se negativo, viceversa (diminuisce la probabilità di far giocare un'azione alla squadra di casa, quindi contestualmente aumenta la probabilità di far giocare un'azione alla squadra in trasferta).

Successivamente, sempre nel metodo init, dopo aver fatto il controllo sul rapportoForza, verrà fatto il controllo se la squadra che sta attaccando, ha un totale di attacco maggiore del totale di difesa della squadra che sta difendendo o meno. In base a questo, verrà precaricata la coda, con degli eventi dipendenti dalle probabilità assegnate:

un esempio può essere il caso in cui in cui la squadra di casa ha rapportoForza>2, di conseguenza ha più probabilità di giocare un'azione e che ha totale attacco maggiore del totale difesa della squadra in trasferta:

```
if(this.rapportoForza>=2) {
    //casa ha il doppio della probabilità di iniziare un'azione rispetto alla squadra in trasferta
    double azioneCasa = Math.random();
    if(azioneCasa<=(0.66+(this.intesa/100))) {
        //attacco squadra casa
        if(this.casa.getTotAttacco()>this.trasferta.getTotDifesa()) {
            //più probabile goal, rigore o espulsioneTrasf
            double prob = Math.random();

            if(prob<0.50) {
                this.queue.add(new Event(1, EventType.PALLA_PERSA, casa));
            }
            else if(prob<0.8) { //30% probabilità di gol
                this.queue.add(new Event(1, EventType.GOL, casa));
            }
            else if(prob<0.85){
                this.queue.add(new Event(1, EventType.ESPULSIONE, trasferta));
            }
            else if(prob<0.9) {
                double inf = Math.random();
                if(inf<0.5)
                    this.queue.add(new Event(1, EventType.INFORTUNIO, casa));
                else
                    this.queue.add(new Event(1, EventType.INFORTUNIO, trasferta));


            }
            else {
                //rigore
                double rigore = Math.random();
                if(rigore<0.8)
                    this.queue.add(new Event(1, EventType.GOL, casa));
                else
                    this.queue.add(new Event(1, EventType.PALLA_PERSA, casa));


            }
```

Gli eventi principali sono sempre PALLA PERSA, GOL, ESPULSIONE, INFORTUNIO. Può verificarsi anche il caso di un rigore. Ad ogni evento, secondo le condizioni citate in precedenza, è assegnata una certa probabilità.

Successivamente, una volta precaricato il primo evento in coda, verrà chiamato il metodo run() per processarlo mediante il processEvent(e).

Al verificarsi di un evento, viene successivamente precaricato in coda l'evento successivo nello slot=slot+1. Anche in questo caso, è tutto fortemente dipendente dalle probabilità assegnate. Se il contatore delle azioni giocate coincide con il numero totale di azioni della partita calcolate in precedenza, nessun ulteriore evento verrà caricato in coda, e la simulazione è terminata. Ad ogni evento, vengono puntualmente aggiornati tutti i contatori nella classe Statistiche, e anche gli eventi della classe MinEvento.

In caso di pareggio, il simulatore decreterà il vincitore ai calci di rigore in base a un parametro che ha ogni giocatore chiamato 'mentalitaRigori'.

La classe Statistiche è composta da tutti i contatori e da una lista di eventi (MinEvento) della partita.

La classe MinEvento è composta da un intero che è il minuto in cui si è verificato l'evento, da una stringa che lo descrive, e dal giocatore protagonista in quel momento. Tale giocatore, viene determinato in base a certi valori:

- In caso di gol, il giocatore che possiede valori di attacco maggiori di tutta la squadra, ha maggiore probabilità di segnare rispetto agli altri.
- In caso di espulsione, il giocatore che possiede valori di aggressività maggiori di tutta la squadra, ha maggiore probabilità di essere espulso rispetto agli altri.
- In caso di infortunio, ogni giocatore ha la stessa probabilità di infortunarsi

Il minuto in cui avviene l'evento, noto lo slot temporale in cui ci troviamo, vengono calcolati nel seguente metodo:

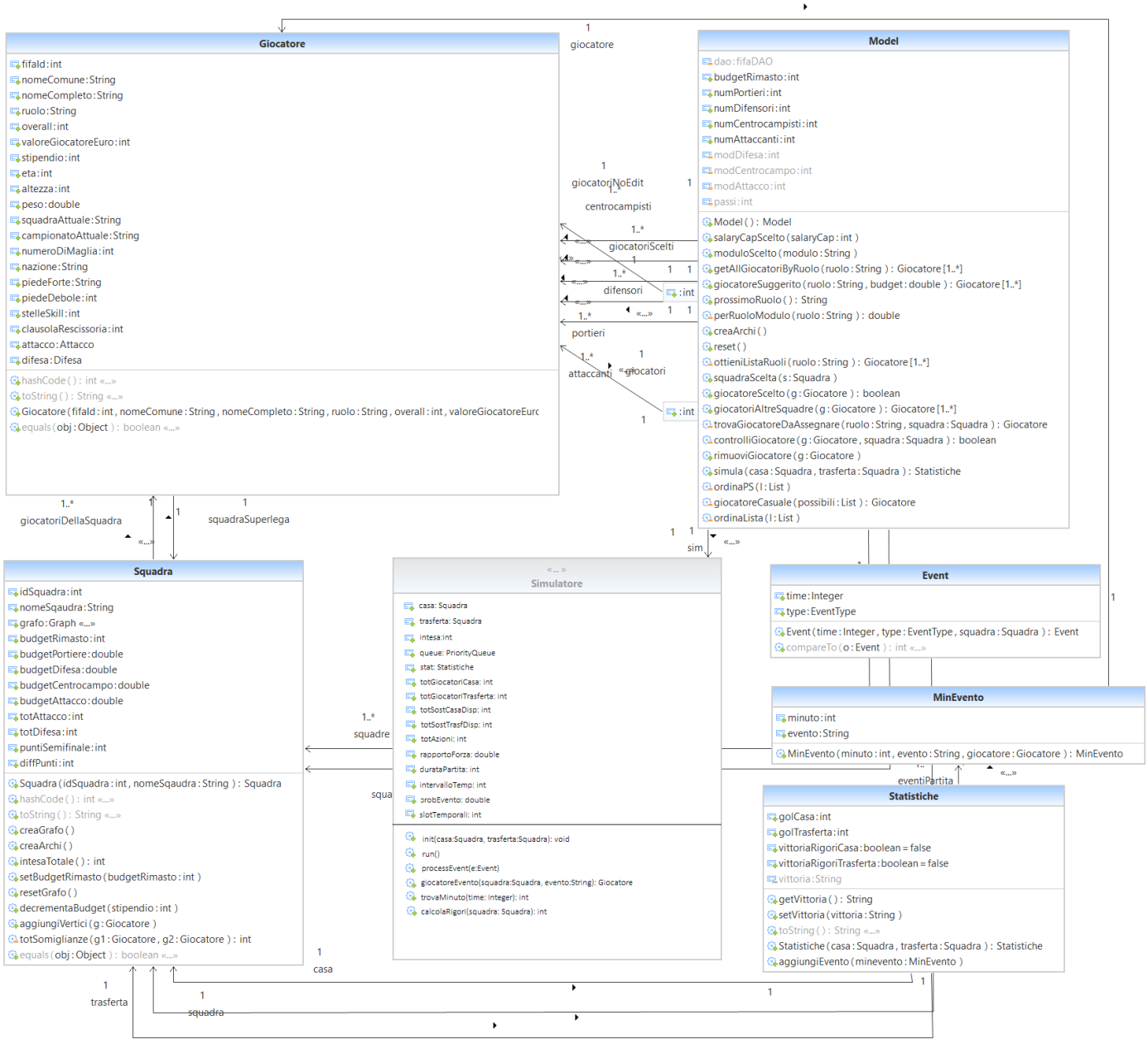
```
private int trovaMinuto(Integer time) {
    int moltiplicatore = this.durataPartita/this.totAzioni;
    int limite = time*moltiplicatore;

    double prob = Math.random()*moltiplicatore+1;
    return (int) (limite-prob);

}
```

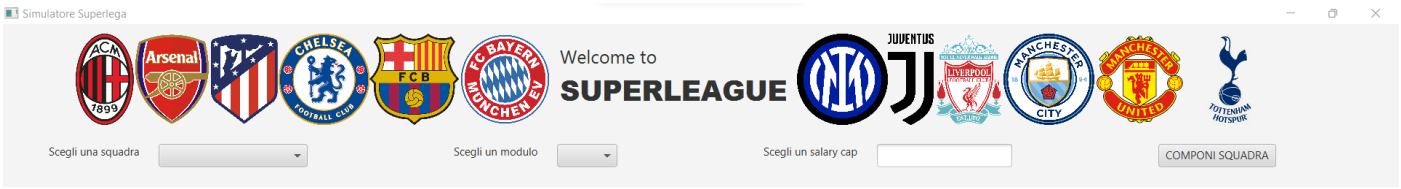
DIAGRAMMA DELLE CLASSI

All'interno di questo schema, sono presenti solo le classi fondamentali per la realizzazione dell'applicativo.

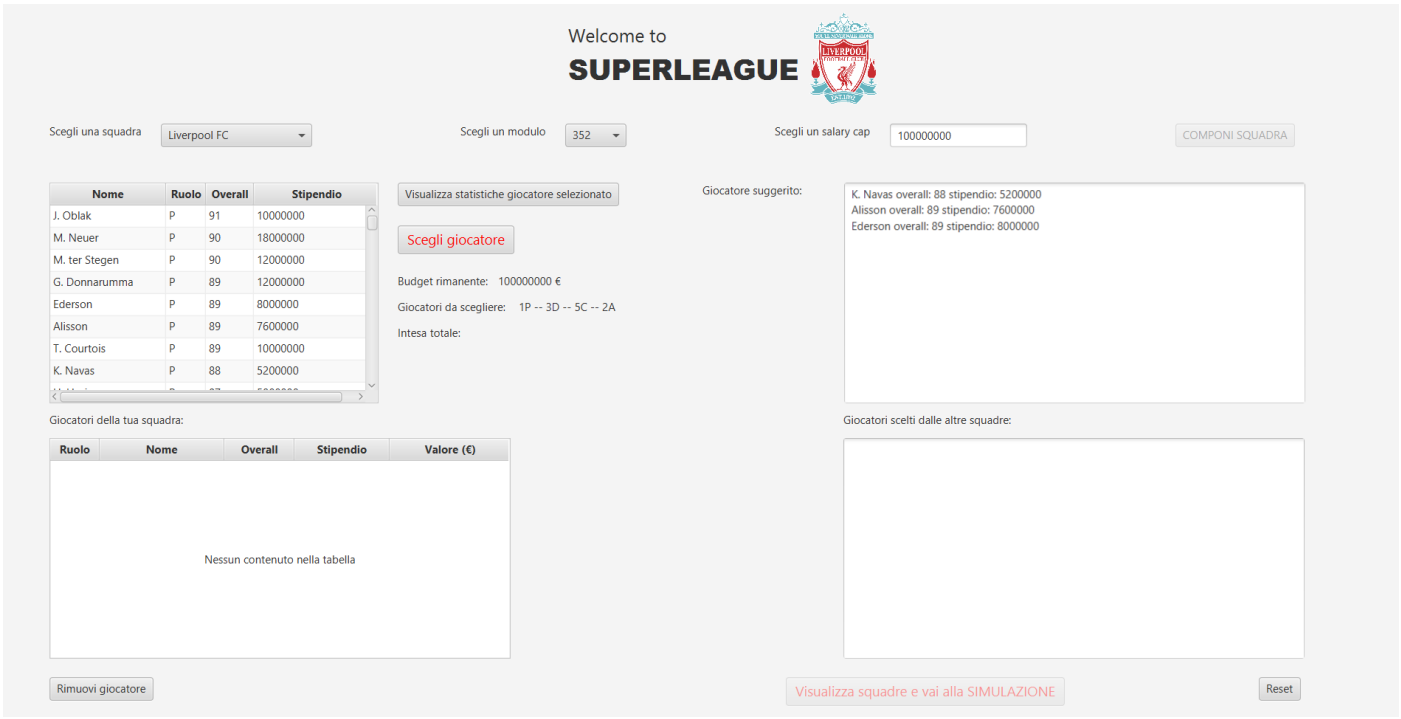


VIDEATE DELL'APPLICAZIONE

All'avvio l'utente deve scegliere una squadra, un modulo e un salary cap.

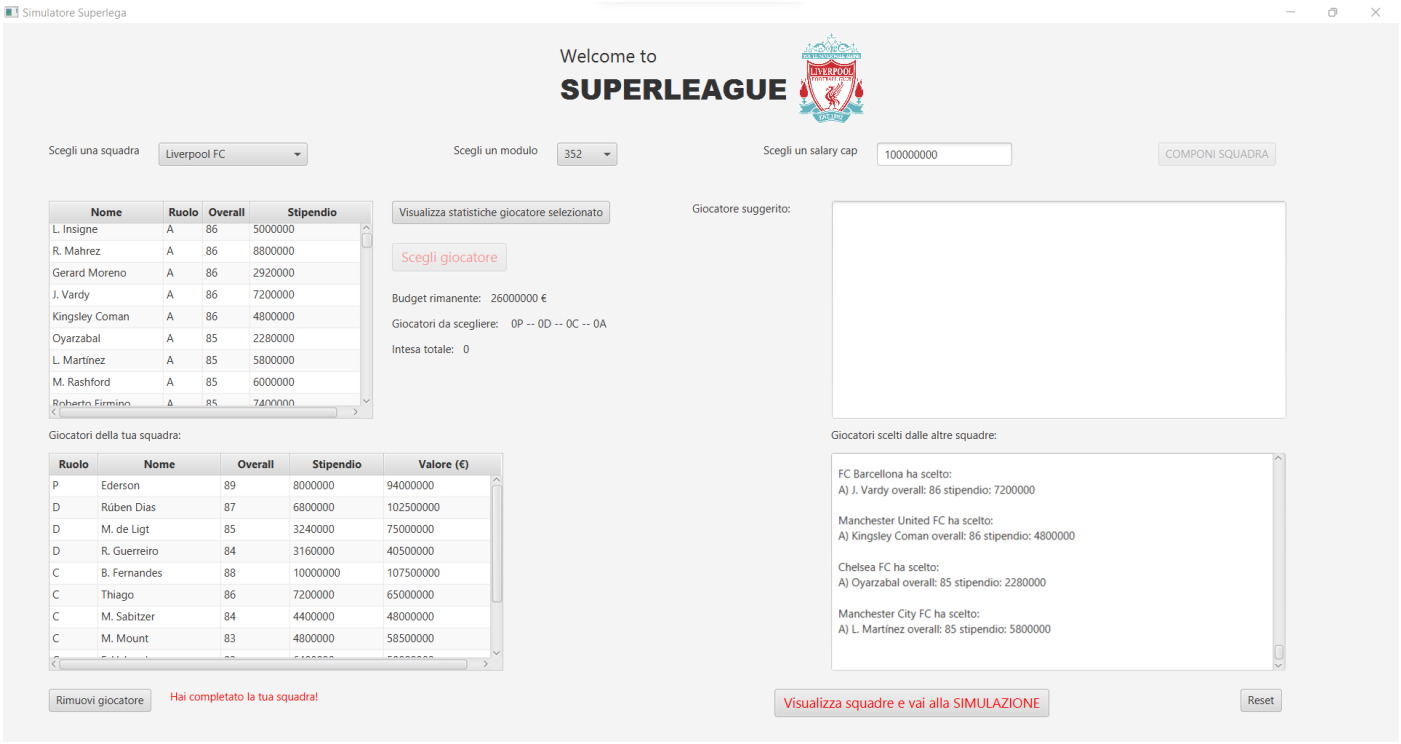


Effettuata la scelta iniziale, potrà iniziare la fase di composizione della sua squadra. Il programma ogni volta suggerisce un giocatore all'utente.



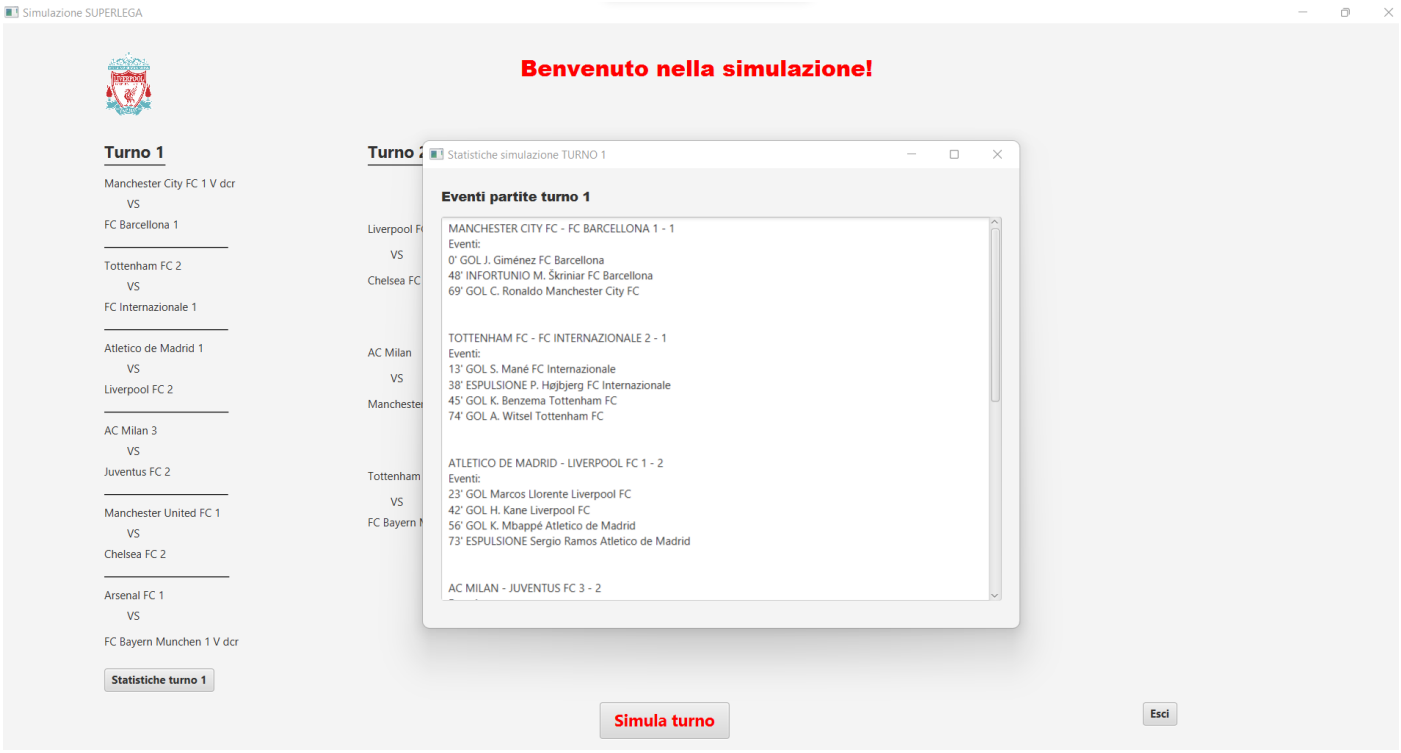
Selezionando un giocatore di interesse dalla table view in alto a sinistra, sarà possibile visualizzare tutte le statistiche del giocatore in questione.

Terminata la composizione della propria squadra, la table view e la text area di riepilogo, saranno completamente popolate. Comparirà un opportuno messaggio di completamento della squadra.



Successivamente, si accederà ad una finestra di riepilogo delle squadre e dei loro giocatori.

Infine, si accederà alla parte di simulazione. Per ogni turno simulato, si potrà accedere al riepilogo degli eventi avvenuti in ogni partita di quel turno specifico.



L'applicativo si conclude, con la visualizzazione del vincitore del torneo.

Link al video di presentazione: https://youtu.be/xRJOOIdsLgU

RISULTATI SPERIMENTALI

A seguito di alcuni test, posso affermare che i tempi di esecuzione degli algoritmi utilizzati sono eccellenti, nella tabella sottostante riporto alcuni risultati.

Campo di test	**Tempi(s)**
Settaggi iniziali	0.0093477
Controlli giocatore scelto e scelta giocatori altre squadre	0.0458138
Visualizzazione statistiche giocatore selezionato	0.33883819
Rimozione giocatore	0.0015485
Giocatori suggeriti	0.0113259
Simulazione turno 1	0.0071224
Classifica simulazione semifinale	6.76399E-4

CONCLUSIONI

Le conclusioni sono più che soddisfacenti, il programma risulta di facile intuizione anche a persone meno ‘esperte’, con tempi di esecuzione eccellenti.

A seguito dei numerosi test, ho osservato che il fattore randomico della simulazione rende i risultati estremamente imprevedibili, anche se con dietro una certa logica.

Mi ritengo estremamente soddisfatto del percorso svolto nell’ultimo anno nel campo informatico, anche a seguito del corso di Tecniche di programmazione, e ovviamente del lavoro svolto per l’elaborazione di questo progetto.

Per i motivi sopra elencati, in conclusione, porgo i miei più cari ringraziamenti al Professor Corno.

Matteo Busnelli